UNITED STATES PATENT APPLICATION

REPLICATING KERNEL DATA THAT IS READ MOSTLY AND WRITE RARELY

INVENTOR

Stephan Gipp
Of
Inver Grove Heights, MN U.S.A.

Aaron Forest Godfrey
Of
Eagan, MN U.S.A.

Schwegman, Lundberg, Woessner, & Kluth, P.A.
1600 TCF Tower
121 South Eighth Street
Minneapolis, Minnesota 55402
ATTORNEY DOCKET NO. 01376.728US1

REPLICATING KERNEL DATA THAT IS READ MOSTLY AND WRITE RARELY

TECHNICAL FIELD

This document relates generally to multi-processor computer systems and in particular to a system and method of replicating data on multiple computing nodes within a computing system.

BACKGROUND

Multi-processor computer systems include a number of processing nodes connected together by an interconnection network. Typically, a processing node includes one or more processors, a local memory, a cache memory and an interface circuit connecting the node to the interconnection network. The interconnection network is used for transmitting packets of information between processing nodes.

In computer systems it is important to minimize the time necessary for processors to access data. In a distributed memory system, communication costs in reading data from remote memory locations can be excessive. To solve this problem, computer memory systems generally use a memory hierarchy in which smaller and faster memories are located within a few machine cycles of the processors and larger and slower memories are located a larger number of machine cycles away. Cache memories are smaller, faster memories that contain a copy of memory data that is used more often by the processors. Data in a cache memory is stored in memory blocks that contain both the data and a tag that identifies the data. If the desired data is not located in cache memory, a cache miss occurs and the data is fetched from local memory.

However, there isn't always enough room in each local memory to store all global variables. Higher communication costs are incurred if a global variable for use on a first processing node resides in local memory on a second processing node. What is needed is an improved method of data access in a multi-processor node system to optimize system performance.

SUMMARY

This document discusses a distributed computer system and a method for replicating data in a distributed computer system.

The system includes a plurality of processing nodes. A processing node comprises at least one processor and at least one local memory, where the local memory is in communication with each processing node. The system also includes maintenance software, to determine whether data is read substantially more often than it is written and to replicate the data that is read substantially more frequently than it is written among the plurality of processing nodes.

The method includes reviewing classes of data, identifying whether at least a portion of data of a certain class used by the processing nodes is read substantially more frequently than it is written and replicating copies of the data of that class in the local memories.

This summary is intended to provide an overview of the subject matter of the present application. It is not intended to provide an exclusive or exhaustive explanation of the invention. The detailed description is included to provide further information about the subject matter of the present patent application.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings like numerals refer to like components throughout the several views.

FIG. 1 shows an embodiment of a multi-processor computer system.

FIG. 2 shows an embodiment of a multiprocessor computer node.

FIG. 3 shows an embodiment of a method for replicating data in a multiprocessor system.

DETAILED DESCRIPTION

In the following detailed description, reference is made to the accompanying drawings which form a part hereof, and specific embodiments in which the invention may be practiced are shown by way of illustration. It is to be understood that other

embodiments may be used and structural changes may be made without departing from the scope of the present invention.

This detailed description discusses a system and method for distributing data to multiple computing nodes in a distributed computer system. The computing nodes typically include processors, cache memories and a local memory.

FIG. 1 shows one embodiment of a multi-processing node computer system 100. The computing nodes 110, 150 include processors 120, 160, cache memories 130, 170 and a local memory 140, 180. In one embodiment, the local memories 140, 180 comprise a global address space. Cache memory 130 is dedicated to the processor or processors 120 in node 110. If processor 120 needs data residing in node 150, processor 120 retrieves the data from the local memory 180 in node 150. Likewise, if processor 160 needs data residing in node 110, processor 160 retrieves the data from the local memory 140 in node 110. If data is identified that is substantially read more frequently than it is written, the data can be replicated and placed in the local memories 140, 180. When the replicated data 190 is needed by a processor, the processor accesses the node local memory 140, 180 and does not need to wait the extra machine cycles for an inter-node access.

If cache memory is used for data access between processing nodes 110, 150, or inter-node accesses, communication cost in handling cache traffic can greatly impact a computer system. Consider, for example, a computer system that includes many processing nodes 110, 150 where each processing node includes multiple processors. If processors 120, 160 are allowed to execute inter-node data accesses directly to the cache memories 130, 170 in other nodes, broadcasting data among the many nodes to maintain cache coherence will negatively impact system performance. For this reason, inter-node accesses are often made through the local memories 140, 180. System performance is negatively impacted if processing nodes have to execute a large number of machine cycles to access data residing the local memories of other processing nodes. If it is determined that a number of these accesses are to data that is substantially read more frequently than the data is written, machine cycles can be saved if this data is replicated 190 among all the local memories by eliminating the extra machine cycles needed to access the local memory of other processing nodes.

One embodiment of data that falls within this class is data written as a result of an external event, such as a result of a system operator changing a computer system parameter. Another embodiment of this class of data is data written automatically, but as a result of a rare event such as a processor going down. Examples of the embodiments include system tunable parameters such as how much extra processor time applications are granted once a processor time limit was exceeded, address lookup tables that allow one processor to look up data specific to another processor such as processor run-queue data, and configuration data written at system boot-up time such as host names. In one embodiment, the replicated data 190 includes data structures. In another embodiment, the replicated data 190 includes portions of data structures.

FIG. 2 shows an embodiment of a multiprocessor computer node 200. The node includes sixteen processors 120. Four of the processors 120 share a cache memory 130. The node 200 also includes local memory 140. In another embodiment, the cache memory 130 includes four cache memories shared by four processors 120. Thus, in the embodiment, the multiprocessor node includes a total of sixteen cache memories 130. In a computer system having many multiprocessor nodes 200, the local memories are in communication with each other through an interconnection network 210.

When the replicated data 190 needs to be updated, the updated data is distributed, or broadcast, to all of the local memories 140 of the nodes where the replicated data 190 resides. In one embodiment, the replicated data 190 resides only on a portion of the local memories 140 in a multi-node computer system 100.

To distribute the data, the computing system 100 includes maintenance software to write data to remote locations. In one embodiment, the maintenance software is included with an operating system running on one or more of the nodes. In another embodiment, the maintenance software is included with applications that run on the system. In yet another embodiment, the data that is to be replicated 190 is identified before an application executes, and the maintenance software replicates the data 190 before such applications execute on the system 100. In yet another embodiment, all of the data 190 to be replicated is not known beforehand, and the maintenance software recognizes data that is read substantially more often than it is written and replicates the data 190 as an application is running.

FIG. 3 shows an embodiment of a method 300 for replicating data in a multiprocessor system that includes multiple processing nodes in communication with local memories. At 310, classes of data are reviewed. At 320, it is identified whether at least a portion of data of a certain class used by the processing nodes is read substantially more frequently than it is written. At 330, copies of the data of that class are replicated in the local memories. In one embodiment, determining whether data is read substantially more frequently than it is written is performed at an operating system level as applications are running. In one embodiment, replicating the data includes executing applications in one or more processing nodes to replicate the data.

Although specific examples have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement calculated to achieve the same purpose could be substituted for the specific example shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is intended that this invention be limited only by the claims and their legal equivalents.